CHAPTER 1

Getting Started

Microsoft has worked for a long time on the latest upgrade to the Windows NT line—Windows 2000. The reasons for the long development process are many and varied, but the one new feature that created the greatest delay also has the greatest potential to change the way developers perform certain tasks. This feature is Active Directory. You'll find that Active Directory will fundamentally change the way both users and developers interact with Windows 2000. Obviously, there are other new features as well, such as Microsoft's DNA Architecture and all the new server types that will come with Windows 2000. This book will introduce you to a lot of programming techniques that you can use anywhere and with any version of Windows. We'll also spend a lot of time looking at Windows 2000's specific features. You'll learn about all the new features that Microsoft has added to the Windows NT legacy.

This chapter concentrates on a lot of the preliminary information that you need to know in order to work with the rest of the book. The first topic covered is the various programming arenas that are encountered when working with Windows 2000. Of course, one of the major areas that we'll focus on is desktop applications, but that's really only the start, not the complete picture. It's important to understand that Windows 2000 can meet more than one need—it covers a lot of different types of programming situations and user needs.

Microsoft has recently introduced the term *Distributed interNet Applications* (DNA) Architecture to describe their vision of future application development. However, although the term is new, the technology it refers to isn't. You've seen many of these players before under different names. The DNA Architecture provides a way of simplifying Microsoft's message to developers, reducing the complexity of Microsoft product offerings, and provides a road map that developers can follow when creating applications. In the next section of the chapter, we'll look at how Microsoft plans to accomplish this goal.

In This Chapter

Understanding the various programming environments

Working with Microsoft DNA

Understanding the DNA Framework services

Working with the DNA Framework tools

Insights into DNA Framework server applications

Understanding the latest changes in Visual Studio

Creating a fully functional client workstation setup

Creating a test server setup

Knowing about your tools is an essential part of creating applications in any environment — Windows 2000 is no different. The tool of choice for this book is the Enterprise Edition of Visual C++ 6.0. Although you'll be able to create many of the examples used in the book using other versions of Visual C++, the screen shots, code, and procedures will all reflect the results of use with Visual C++ 6.0. In the third section of the chapter, I'll tell you what's changed with this version of the programming language so you can decide whether or not you need to upgrade.

In the last two sections of the chapter, we look at an essential element of development — the workstation and server setup. Yes, you can develop applications just about anywhere, but it pays to have a system setup that reflects the experimental nature of programming. You should never develop applications on an existing production system because the development code you're creating could fail. In addition, this book makes certain assumptions about your workstation and server setup. Knowing what those assumptions are will help you greatly when it comes time to work with the sample applications.

Windows 2000 Programming Arenas

Windows 2000 is a large operating system. Some industry leaders have said that this operation system is at least the size of a minicomputer operating system. Add to the size and complexity of the operating system, the size and complexity of today's development tools, and you have a real chance of getting lost before you even start writing code. It's important, therefore, to consider just what kind of project you're going to create before you actually begin to create it.

I'm not talking about a specification here, but rather a simple preliminary overview of what you want to do. After all, if you begin writing the application specification before you have a good overview of the project, you're apt to get lost by not being able to see the forest for the trees. Just as an author starts with an idea, then moves to an outline, and finally begins writing the book itself, today's developer has to get a good overview of what kind of application would work best for a particular need.

At this point, some of you may be wondering just where this is all leading. Consider a situation in which you need a simple utility application that helps a salesperson calculate the total for an order (including elements like shipping, handling, and tax). You could write a standard Windows application to do the job. However, when you consider that the salesperson is likely to be on the road most of the time, it might actually make more sense to add this application to the company's Internet connection so that the salesperson could access it from the road.

As you can see, it might not be easy to select an application type with all of the choices available today. In the following sections, we are going to look at some additional comparisons — arenas that require an overview before you can proceed with a software specification.

Desktop: Win32 versus DHTML Clients

The browser has become more and more powerful, and the tools used to work with it have largely kept pace — at least when it comes to the presentation of text. In fact, Internet Explorer is nearly as large as some operating systems today. You can use Internet Explorer to display both local and remote files with equal ease and, because of the object technology that many browsers include, the files you see don't necessarily have to be HyperText Markup Language (HTML) pages.

It's little wonder that Microsoft has chosen to use HTML as the media for help files for Windows and several of its other products. However, HTML provides only static displays. This makes plain HTML more useful for things like help files than it would be for an active application like a stock ticker. That's where Dynamic HyperText Markup Language (DHTML) comes into play. With DHTML, you can now provide a presentation that changes with time — keeping the information presented fresh and up to date.

So now programmers have to decide which technology they should embrace; the standard Win32 application or the newer DHTML application? Obviously, there are several things the developer will need to consider. For one thing, Win32 applications tend to run faster, and they're still better by a long shot when it comes to performance. On the other hand, DHTML applications are more portable, and they aren't restricted to the desktop.

From a development perspective, you'll find that each class of application has some potential problems. The development curve for desktop applications has taken a definite upswing in recent years simply because users demand more and the tools for creating complex applications have become better. DHTML applications require knowledge of multiple programming languages (JavaScript macros *do* count as programming), making them more difficult for the developer to create from a learning curve perspective. In short, there's no free lunch — desktop application development is complex, and you need to consider which route will provide the best result in the least time in your particular situations.

You'll get a lot of help with both of these application types later in this book. Chapter 3 will give you an overview of the various desktop application types and when you would use them. Chapters 4 through 6 will help you get beyond desktop basics. You'll find the latest information about DHTML applications in Chapter 7.

Server: Web Server versus COM+ Business Components

Getting the vast stores of information on your server formatted and distributed to the client can be a problem even in the best of situations. However, today's corporate environment is a mishmash of competing standards, old and new

equipment, applications for which no one can find the source, and ever-increasing expectations on the part of users. It's a wonder that the average developer of corporate applications hasn't gone completely crazy by now.

Microsoft provides two basic ways of answering the needs of server development today, though those two routes take a variety of forms. The first method is to use a Web server to distribute information. The advantage of this method is that anyone can access the information anywhere (unfortunately, this also brings the specter of data loss and other security problems). The second method is to use COM or COM+. The advantage of this method is that you can create LAN and WAN applications that actually rely on both old and new code (in many cases, COM can be used as a wrapper for old applications).

Of course, the most likely scenario is that you'll use a combination of these methods to respond to your server programming needs. There probably aren't any companies that are so Web-enabled that they could rely exclusively on the Internet and local intranets for company-processing needs. On the other hand, everyone is getting onto the Internet because it provides such great accessibility features.

We're going to be looking at a wealth of different server solutions in this book. If you're interested in the Internet side of things, check out Chapters 8 through 10 where everything from ISAPI to Active Server Pages (ASP) scripts is covered. On the other hand, if you're more interested in Microsoft's COM+ offerings, you'll find what you need in Chapters 11 through 16. These chapters will cover everything from simple COM components to more complex Microsoft Message Queue (MSMQ) and Microsoft Transaction Server (MTS) solutions.

Intranet and Internet

Developing for the Internet is not quite the same as developing for a local intranet. Sure, there are a few similarities; you're still producing some type of DHTML code that will get displayed for the user within the confines of a browser, but some of the principles behind the two development environments differ.

Consider the problem of security for a moment. An intranet application is inherently more secure because no one outside of the company (except for business partners) will normally see the application. All you really need to do is verify the identity of the user at the beginning of a session and encrypt the data if the application will transfer information outside of the building. On the other hand, Internet applications require some method for maintaining individual user information confidentiality, as well as ensuring that no one else is breaking into the connection between client and server. Obviously, data encryption is a must, in this case, and not just a nicety that can be added when needed.

We'll take a special look at the security needs of various application types in this book, along with the special features of Windows 2000 that makes keeping your data secure a lot easier. Make sure you look at both Chapters 17 and 18 for more details on the security provisions of Windows 2000.

Large Project

The final kind of application isn't really unique, but it does bring with it unique management requirements. A large project is really just a combination of multiple small projects. A three-tier application is really just a combination of a desktop, business component, and database application in disguise.

Coordination of resources becomes a problem with large projects because everyone will want to use everything. The client portion of the application will require access to the server to fully test the results of various business logic calculations, but the business component development team will need access to the server as well. In other words, resource coordination will be your first concern.

There's also the problem with the chicken-or-the-egg syndrome. Deciding which part of the project to start with can be a real nightmare. In some cases, you'll definitely need parallel development to get the job done in a reasonable amount of time. However, figuring out when it's time for parallel development can be a problem — start too soon and all you'll do is provide a means for everyone to get in everyone else's way.

As you can see, large project development is not something you should take on before you've gotten a few simple projects under your belt. We will cover all of the basics steps of a large project in this book, but we won't cover the additional management concerns that a large project usually requires. However, you'll definitely want to spend some time finding out about these large project-management concerns before you tackle your first large project.

The Microsoft DNA Architecture

There's no doubt that Microsoft is a large organization that develops a lot of different application types, including a variety of tools for developers. In fact, the very largeness of Microsoft has become a problem in recent years because no one is really aware of how many products they produce. As a developer, I'm usually swamped trying to read up on the latest Microsoft technology, only to find out that there are four new technologies that I haven't even begun to learn about.

Part of the reason for the creation of the Microsoft Distributed interNet Application (DNA) Architecture is to reduce the perceived complexity of the myriad of Microsoft offerings. DNA seeks to bring some order out of the chaos of today's development environment, at least from a Microsoft perspective. One way to think about DNA is as Microsoft's version of a road map for application development.

However, to stop learning about DNA at this point would be like stopping after reading the first sentence of a book and declaring you know the rest of the story. DNA has a lot more to offer the developer than simple product advertisement (although the sales pitch angle is definitely there for all to see). DNA is also a framework, a way of dividing the complex development environment of today into more manageable pieces. DNA encourages the team-development approach, helps development teams understand where process boundaries occur, and helps management create a phased-development approach when necessary.

DNA is also an extension of the client-server technology that has been used in many development situations to date. However, the development scenario is now divided into three tiers (navigation and user interface, business logic, and data storage) instead of two, and DNA incorporates the Internet into the development environment. In other words, DNA isn't more of the same rhetoric, though parts of the story will definitely sound familiar.

According to Microsoft's vision of the computing world, DNA also provides you with the capability to produce a whole suite of new applications types. The most important applications include electronic commerce and intercorporate communications. Obviously, there are other tools on the market that will allow you to develop these kinds of applications, but DNA probably provides the most encompassing technology for developing a wide variety of these application types. (This is one of the few advantages to the sheer number of product offerings that Microsoft provides.)

There are certain cost savings that you'll experience when using DNA as well. The most obvious cost savings is development effort. Sure, it takes just as long, perhaps longer, to create an application using DNA. However, once you develop the application, you can use it anywhere. You no longer have to produce different versions for each environment in which the application has to execute.

Another potential cost savings is in management costs. Because of the Internet orientation of DNA Architecture applications, all applications can be managed from a central location, making administrators much more efficient (and less frustrated). The costs incurred after an application is in place are constant, so even a small savings here will amount to a large savings over time. Part of the management savings comes in the form of application changes. Older applications require the administrator to perform a desktop-by-desktop installation. Theoretically, you can make most required changes to a DNA Architecture application by updating server components — again, this represents a major cost savings for companies.

Now that you have a better idea of what DNA is all about, let's take a closer look. The following paragraphs will help you put the different DNA pieces together. We'll look at the various components and what they mean for you as the developer.

DNA for Horizontal and Vertical Markets

Many people were initially skeptical of Microsoft's DNA Architecture. The main reason is that Microsoft didn't appear to be putting much support behind the project. However, in the months that have followed since that initial introduction, Microsoft has made DNA a central part of its message, especially when it comes to development.

Of course, most of Microsoft's development efforts are horizontal (general) market oriented. For example, all versions of Windows are directed at the horizontal market. In fact, nothing in Microsoft's literature or conferences like the Professional Developer's Conference (PDC) would lead you to believe there's any vertical market emphasis by Microsoft at all.

However, Microsoft is developing specialty versions of its DNA architecture for the vertical market. For example, Microsoft just recently unveiled plans to create a special version of DNA for the manufacturing market known as DNA-M. Several vendors have already signed on for the program, including Nike(tm), Ocean Spray(tm), and Clorox(tm).

The thing that makes the DNA-M version of Microsoft's Architecture different is that it includes elements that are manufacturing specific. It concentrates on technologies like Windows CE, which will allow people on the manufacturing floor to provide information to the company as a whole using handheld computers. The entire DNA-M suite will extend from the shop floor to enterprise resource planning (ERP).

One thing that DNA-M won't provide is actual automation technology. In fact, two of the supporters for DNA-M, Honeywell and Rockwell Automation, are also providers of automation equipment. What Microsoft is trying to provide is support for a manufacturer's existing automation, so the vendor can make better use of the technology at hand. The ability to integrate the various elements of a manufacturing environment is the main feature that DNA-M will provide for manufacturing companies that implement the technology. Some of the companies who are using DNA-M now report a 30 to 50 percent savings in development time, which is significant in any environment.

There are also other special versions of DNA for other vertical markets. Microsoft has plans to provide special versions for the healthcare, retail, life insurance, and financial services vertical markets. If you want to find out more about vertical market support, check out Microsoft's Web site at: `http://www.microsoft.com/dna/default.asp`.

Services

Creating a new framework in which to develop applications means that the vendor also has to supply any required services needed to implement that framework. Microsoft has done just that by augmenting the capabilities of Windows 2000. Though these new services will probably require some level of tweaking before Microsoft gets them right, you'll find that they do make the idea of DNA more realistic. The following list provides you with an overview of many of the new services provided by Windows 2000 from a developer perspective (we'll cover many of these services in more detail as the book progresses).

Table 1-1
Windows 2000 New Services Overview

Service	*Description*
Security	As previously mentioned, communication over the Internet requires a higher level of security. You can't assume that the security measures that worked fine on a LAN are going to come even close to working on the Internet where every cracker has open access to your lines of communication. You'll find that Windows 2000 provides a wealth of new security features, including Kerberos and Public Key Infrastructure (PKI).
Directory	An all-encompassing framework requires an equally impressive directory structure. Active Directory is Microsoft's entry into the new or improved directory race. Novell has a decided advantage in this particular area right now, especially from a network administrator perspective. It truly is easy to manage network objects using the Novell product, and there isn't any chance that Active Directory will provide the level of interoperability that NDS currently does anytime in the near future. However, as you'll learn throughout this book, there are some decided advantages for the developer in using Active Directory rather than Novell Directory Services (NDS). For example, Active Directory will allow you to create global registry settings that the user will download when he or she logs onto a machine—a decidedly better way than updating those settings on a desktop-by-desktop basis as network administrators have to do now.
Transactions	Microsoft Transaction Server (MTS) enables you to create a safe environment for your data by ensuring that all updates for a given operation are completely finished or rolled back. In other words, you don't have to worry about only half of the data in a form making it into the database. Either the entire form will make it, or none of the form will. This means that you can check for the success of an operation and take appropriate action based on what you find with greater ease and less chance of error.

Service	*Description*
Message Queuing	Microsoft Message Queue (MSMQ) enables you to do a number of new things. One of the more interesting ways to use this server is for disconnected applications. For example, a salesperson can go on the road, make all of theirhis or her sales, and update the central database, even if they don'tthe or she doesn't have a telephone connection. From the user's perspective, the application works just as if it did have a connection to the central database. The difference is that all of the transactions are stored as messages. When the user does make a physical connection to the office (either through phone or a LAN connection), the central database gets updated.
Database Management	Windows 2000 provides a number of new database management capabilities. Some of these capabilities are associated with products like MTS and MSMQ, which allow for disconnected application scenarios. However, you'll also find some type of database operations are sped up through the user of new technologies like In-Memory Databases (IMDB), which are small lookup tables that reside in memory at all times.
Data Access	Several technologies make creating connections with DNA easier than ever before. For example, Microsoft Data Access Components (MDAC) version 1.5 provides the functionality required to create data-bound pages for Internet Explore versions 4.0 and above. You'll also find that both ActiveX Data Objects (ADO) and Remote Data Services (RDS) provide improved methods for creating a remote connection—even if that connection is over the Internet. Most of the data-access technologies were in place before Windows 2000 came along. However, it's important to realize that Microsoft has improved support and made these technologies standard for Windows 2000, ensuring that they'll be available when you need them.
E-mail and Collaboration	Currently, when you send an e-mail message, it usually contains text or perhaps some type of attachment. In most cases, that e-mail is retrieved by a human who expects its arrival. What would happen if you could send objects via e-mail rather than just text or attachments? The ability to send objects is provided by MSMQ. It's a powerful feature that will enable application developers to create scenarios where entire objects get moved from one place to another with the same ease as sending e-mail.

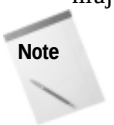
Continued

Table 1-1 *(continued)*

Service	*Description*
Scripting	Windows 2000 provides two forms of scripting that will interest developers. The first is an improved form of Windows Scripting Host that makes it easier for developers to create scripts that allow for the completion of tasks well beyond the batch processing that developers used in the past. The second form of scripting is ASP. You'll find that you can create even more complex Web pages than before by using DHTML (and perhaps by the time you read this, XML).
Systems Management	Previous versions of Windows relied exclusively on monolithic management applications. If you wanted to change a user name, you had to load the User Manager. Likewise, changes in your Internet Information Server (IIS) setup might require a visit to the DHCP Server Manager program. Windows 2000 uses the Microsoft Management Console (MMC) for management tasks. MMC enables you to create snap-ins that customize its features. This means that you can create special management applications for network administrators that enable them to completely manage a DNA Architecture application from one location.

Tools

Just having the services required to get a DNA Architecture application off the ground isn't really enough to make it a reality. You also need the tools to get the job done. It's hard to say just when Microsoft will provide a one-package solution to all of your DNA problems. Today, you'll need a minimum of one Microsoft programming language product and the Platform Software Development Kit (SDK) to actually use most of the Windows 2000 specific features described in this book. Let's hope that by the time you read this, Microsoft will have at least started a beta for the next set of tools required to make DNA a complete reality for developers. In the mean time, the Microsoft products do offer a wealth of features that make DNA development a possibility. The following list provides an overview of some the major features of Windows 2000.



We'll spend some more time looking at these features as the book progresses.

Table 1-2
Windows 2000 New Features Overview

Feature	***Description***
RAD Tools	Rapid Application Development (RAD) seems like an oxymoron in today's development environment. It seems like everyone is talking about the increase in the number of lines of code required to create applications, with no decrease in sight. Fortunately, wise use of Wizards and IDE features can reduce the number of lines of code that you'll actually have to type. In addition, Windows 2000 provides new API functions that will reduce the number of calls you need to make to actually implement a DNA Architecture application.
Component Creation	The business logic portion of the DNA Architecture relies heavily on components. The emergence of COM+ in Windows 2000 makes this particular part of creating an application a little easier. The combination of COM and MSMQ makes it possible to create a component that allows MSMQ to do at least part of the work, freeing you to work on features that are specific to your component rather than general housekeeping chores.
Team Development	DNA is all about team development. In the past, it was hard to define where one part of an application started and another ended. Using the road map provided by DNA makes it easier to divide an application in manageable pieces and improves team development potential. In addition to the road map, Microsoft products like Visual Studio make it easier to create a team development environment. Not only can you use the right tool for the task at hand, but the tools provided in Visual Studio enforce team development concepts like code check out as well.
Content Creation and Publishing	Many, if not most, DNA Architecture applications rely on some form of Web-page output. There are several ways to create this output, but most of them involve spending some time with a Web-page design product. Currently, Windows 2000 ships with Front Page. However, in a team development situation, you'll probably need a heavier duty team-oriented product like Visual InterDev.

Continued

Table 1-2 *(continued)*

Service	*Description*
Web Site Development	The Windows 2000 MMC includes an IIS snap-in that makes administering your Web site much easier. You'll find that you can perform all of the tasks required to set up any number of Web sites, at a single console.

What's New in Visual C++ 6?

As previously stated, the tool of choice for this book is the Enterprise Edition of Visual C++ 6. You can use previous versions of Visual C++, or even another vendor's product for creating the example applications, but it's unlikely that the screen shots you see will match mine, and you may have to rework the code to some extent. (The educational versions of Visual C++ will definitely not have the features required for many of the examples in the book—you'll want to get an update if possible.) Users of the Professional Edition of Visual C++ 6 will be able to create most examples, but will likely have problems working with any database management examples.

Some of you may be new to Visual C++ 6, so it's important to know a little about this product before we get started. Obviously, it would be nice to know what Visual C++ 6 provides that's so different from its predecessors. The following list will help you understand how Visual C++ 6 provides more in the way of developer tools that make creating the newer classes of applications in this book easier.

/GZ Compiler Option—Catch Release-Build Errors in Debug Build

Ever run into a bug that's there in your release build (when you can't access the information required to catch it), but totally absent in your debug build? This compiler option is designed to help you with that problem. It enables you to use the /O1, /O2, /Ox, or /Og optimization switches. Optimization is a source of many release build problems. However, Visual C++ will ignore any #pragma optimization statements in your code, which means that you may not find all optimization bugs. The /GZ compiler option also performs the following three tasks:

- **Automatically initialize local variables:** Visual C++ will automatically initialize local variables to a value of 0xCC. This means you'll be able to find uninitialized variables faster. It also enables you to find variables not normally pointed out with the C4700 or C4701 warning messages.

- **Function pointer call stack validation:** Visual C++ checks each function to make sure it works properly with the stack. The ESP register value is checked on function exit to make sure that it's the same as on function entry. This check enables you to find mismatches between what the called function expects in the way of cleanup and what the calling function provides in the way of cleanup when used to call a function through a function pointer.
- **Call stack validation:** Visual C++ checks the stack pointer (ESP register) at the end of each call to make sure it hasn't changed. This enables you to find errors in in-line assembler routines or mismatches in function calls.

__forceinline Keyword

This is a new version of the __inline keyword provided in previous versions of Visual C++. Placing functions in-line eliminates function calling overhead by placing the called function directly within the calling function. You use it to enhance application speed. Unfortunately, using in-line code also increases code size because the in-lined function will appear within the calling function each time the in-line function gets called.

Visual C++ normally performs a cost analysis of the in-line function to see if it's actually worth the increase in code size to gain the speed advantage of using in-line code. There are two situations in which Visual C++ normally creates a separate copy of the in-line function and calls it separately instead of using in-line code: recursive function calls and functions that are referred to elsewhere in a translation unit.

The __forceinline keyword tells Visual C++ not to perform the normal analysis and to always use in-line code. In other words, the programmer is taking responsibility for any large increases in code size that may result when using in-line code in certain situations such as recursive function calls. As a result, you have to use this new keyword with extreme care to ensure that your code size remains manageable.

There are still some situations in which the compiler still can't force in-line code, even if the __forceinline keyword is used. For example, you can't use in-line code with the /Ob0 command line switch (the normal switch for debug builds). Another situation in which you can't use in-line code is if the called function has a variable list of arguments. Whenever the compiler runs into a situation where it can't force in-line code, you'll get a level 1 warning message (number 4714).

ADO Databinding

Microsoft is constantly trying to improve data access for its programming language products. ADO (ActiveX Data Objects) represents a new way to provide database access through the combination of databound ActiveX controls and an ADODC (ADO Data Control). The ADODC acts as a data source that defines where the

information you want to display is stored and the requirements for accessing that data. You'll need to provide the ADODC with six pieces of information: OLE DB provider name (like SQL Server), DSN (the data source name as specified in the ODBC applet of the Control Panel), user name, password, record source (usually a SQL query), and connection string. The ActiveX controls are used to display the contents of the data source.

ADO provides several advantages over previous database access methods. The following list describes them for you.

- **Independently created objects:** You no longer have to thread your way through a hierarchy of objects. This feature enables you to create only the objects you need, reducing memory requirements and enhancing application speed as a result.
- **Batch updating:** Instead of sending one change to the server, you can collect them in local memory and send all of them to the server at once. This results in improved application performance (because the update can be performed in the background) and reduced network load.
- **Stored procedures:** These procedures reside on the server as part of the database manager and are used to perform specific tasks on the dataset. ADO enables you to use stored procedures with in/out parameters and return values.
- **Multiple cursor types:** Essentially cursors point to the data that you're currently working with. Theoretically, you can even use backend specific cursors.
- **Returned row limits:** You only get the amount of data you actually need to meet a user request.
- **Multiple recordset objects:** Enables you to work with multiple recordsets returned by stored procedures or batch processing.
- **Free threaded objects:** Enhances Web server performance.

There are two databinding models used for ActiveX controls. The first, *simple databinding*, enables an ActiveX control like a text box to display a single field of a single record. The second, *complex databinding*, enables an ActiveX control like a grid to display multiple fields and records at the same time. Complex databinding also requires the ActiveX control to manage which records and fields are displayed, something that the ADODC normally takes care of for simple databinding. Visual C++ comes with several ActiveX controls that support ADO, including the following:

- DataGrid
- DataCombo
- DataList

- ✦ Hierarchical Flex Grid
- ✦ Date and Time Picker

ATL Composite Control

This feature will enable you to create a new ATL (ActiveX Template Library)-based control using the other controls at your disposal. You'll start out with a dialog box that you can fill with Windows and ActiveX controls. The dialog box takes up the entire client area of your new control. You can create an ATL composite control using one of two wizards: ATL COM AppWizard or ATL Object Wizard.

The reason that most programmers will use this technology is to create a form that they can display as a single ActiveX control. For example, if you're currently forcing the user to download a form's worth of controls to display a Web page, you can reduce the complexity of the task using an ATL composite control. The user will only need to download one control once you make the change. Not only will this make the download quicker, but you'll reduce user frustration as well (the whole control will download or it won't, no more partial form downloads).

AutoCompletion

The AutoCompletion feature helps complete statements in your C++ header and source code files for you. What you'll see is a list box with possible completion lines. For example, consider the following code:

```
CString      oMyString;
oMyString.
```

As soon as you type the period for oMyString, you'll see a list box containing the various methods and properties associated with a CString object. At this point, you can simply scroll down to the method or property you want to use and click on it. Or you can type the first few letters of the method or property that you want to use, and the list will scroll to show you methods and properties of that name. Once you do select a method or property, Visual C++ will automatically type the name of the method or property you want to use. The result is that you'll spend a lot less time tracking down typos and spend more time coding.

If you loose the AutoCompletion list box, right-click next to the place it originally appeared and choose AutoComplete from the context menu. The list box will reappear.

Now, let's say that you're looking through the AutoComplete list and aren't sure about the limitations of the AnsiToOem( ) method of the CString object. If you highlight this item and then position your mouse over it, what you'll see is some balloon help telling you there's a 255 character maximum. This form of balloon help is another new Visual C++ feature called Doc Comments.

Not only are Doc Comments part of the Visual C++ package, you can add them to your functions and variables as well. All you need to do is add a comment either before the function or variable declaration, or on the same line as the declaration. You can use this feature to tell someone all about a function or variable that you've provided. Obviously, you don't want to make the comment too large. Microsoft places a 12-line limit on Doc Comments, which should be sufficient for most purposes.

Doc Comments aren't limited to the AutoCompletion list box. Try positioning your mouse over a variable or function. What you'll see is the same balloon help that Doc Comments provides within the AutoCompletion list box. As you can see, Doc Comments is a very fast way to provide continuous documentation for your application and makes it easier for someone else to use the code that you've designed.

ClassView and WizardBar Improvements

Microsoft has provided some new features for both ClassView and WizardBar. The following list tells you about them:

- **Delete member function:** You now have the ability to remove a member function with a lot less work. All you need to do is place the cursor anywhere within the member function, and choose Delete from the WizardBar Actions drop-down listbox or Delete Function on the Message Maps tab of the ClassWizard dialog box. In either case, Visual C++ will remove the function declaration from the header file and comment out the function body from the source code. This allows you to recover the function later if you decide you need it after all.
- **Go to Dialog Editor:** This option is available only when you're creating an application with a dialog box. Select the class containing the dialog box, and then choose the Go To Dialog Editor option from the WizardBar Actions drop-down listbox.
- **New form:** This option adds a new form class to your project. You'll see a New Form dialog box where you'll choose the form name, its base class (the most common being CDaoRecordView, CFormView, CDialog, and CRecordView), and a resource ID. The base class choices that you'll see are partly dependent on the type of application you're creating and any choices you made when using the application wizard. In some cases, you'll also get to choose the level of automation and provide document template information. Visual C++ will gray these options in the New Form dialog box when they are not available.
- **Empty class tracking:** The WizardBar now tracks all classes, even those without member functions. The Members combo box on the WizardBar will display "No Members - Create C/C++ Member Function" if you choose a class without members. The default WizardBar action is to add a new member function to empty classes.

Command-Line Builds

Visual C++ provides a new MSDEV utility that enables you to build projects at the command line without exporting a MAK file first, and then using the NMAKE utility. The syntax for the MSDEV utility is as follows:

```
MSDEV <Filename> [/MAKE "<Project Name> - <Configuration Name>
| ALL"] [/REBUILD /CLEAN /NORECURSE /OUT <Log File> /USEENV]
```

You must provide a DSP (project) or DSW (workspace) filename. The following list describes the optional command-line switches:

- /MAKE: Enables you to define which project and configuration to make. The project name is always the name that you've given the project during the creation process. The configuration name can either be something specific like Win32 Debug or all of the available configurations (using the ALL keyword). Visual C++ allows multiple /MAKE arguments on a single command line, so you can create any number of configurations.
- /REBUILD: Clears all intermediate files before beginning the compile. It then builds the new intermediate files. This ensures that you get a completely clean build.
- /CLEAN: Clears all intermediate files before beginning the compile, but doesn't rebuild them.
- /NORECURSE: Builds the current project, but not any dependent projects.
- /OUT <Log File>: Redirects the output from the screen to a log file. The log file enables you to analyze build errors and correct them.
- /USEENV: Ignores the tools.options.directories settings found in the DSW or DSP file and uses the current environment settings instead. This enables you to override environment settings without reopening Visual C++, but also means that you may introduce new errors.

Compiler Throughput Improvements

Compiling projects with large PCH files will take less time. You'll also notice improvements in smaller projects, but the improvements are most noticeable with projects that use large PCH files.

Delay Load Imports

Normally, there are two choices when working with DLLs; you can automatically load them with the application or you can load and unload them manually as needed. The Delay Loads feature gives you a third option. It allows an application to load a DLL automatically when needed. This means that if a user doesn't require

the services of the spelling checker in your word processing application, that DLL doesn't get loaded.

Using this feature doesn't require much extra work on your part. All that's required is that you add the /DELAYLOAD command-line switch when linking your application and the DELAYIMP.LIB to the list of linked files.

Dynamic HTML

A new MFC class, CHTML, allows you to add a Dynamic HTML (DHTML) view to your application. What this class really gives you is the ability to add a Web browser to your application without a lot of extra programming. Obviously, since this is a Microsoft product, your Web browser will mimic the characteristics of Internet Explorer 4. This feature allows you to mix desktop and Internet applications in a way that is almost transparent to the user.

Dynamic Parsing

ClassView is now automatically updated as you type new class, function, and variable information into your application. This enables you to see changes to your application as you make them rather than when you save the source file. It also means that elements like the WizardBar and ClassWizard are always up to date.

This feature currently has some problems that Microsoft promises to fix by product release (at least as of this writing). For example, certain types of typos can cause Visual C++ to parse what you're trying to do incorrectly and the IDE will freeze as a result. In all cases, restarting the IDE will clear the problem.

Edit and Continue

The edit and continue feature will enable you to make small changes to your application while you're debugging it without rebuilding the project from scratch. Not only will this make debugging faster, but you could perform "what-if" analysis on your application as well.

There are limitations to what you can do with edit and continue. Visual C++ will simply prompt you to rebuild your application if you exceed the bounds of what edit and continue can do for you. The following list provides some general guidelines on places where you can't use edit and continue.

- Header files
- Global/static data
- Class definitions and function prototypes
- Introduction of new variable types
- Exception handling blocks

The edit and continue feature is on by default, so you don't need to do anything special to use it. If you do want to turn the feature off, using the Tools | Options command to display the Options dialog, select the Debug tab and uncheck the Allow Edit and Continue debugging option. You'll need to completely rebuild your project when turning this feature on or off.

Extended Stored Procedure Wizard

Stored procedures enable you to manipulate database data in some way and provide a result as an output to the user. Visual C++ provides a new Wizard that helps you create stored procedures for SQL Server. These stored procedures use a COM interface with a single function with the name of the extended stored procedure. Once you register the extended stored procedure DLL on the SQL Server, you'll be able to access the data that it retrieves using a standard COM interface.

New Debugger Features

Visual C++ provides a number of new debugger features that make it easier to find bugs in your application. Table 1-3 contains a complete list of these new features and provides an overview of each one.

Table 1-3
New Debugger Feature Overview

Feature	*Description*
Auto expansion of VARIANTs and GUIDs	The AUTOEXP.DAT (auto-expand) file contains new rules that allow you to display VARIANTs and GUIDs in more meaningful ways. (The AUTOEXP.DAT file appears in the Program Files\Microsoft Visual Studio\Common\MSDev98\Bin directory on your hard drive.) For example, a VARIANT will now appear as the data type that it actually contains; a string will actually look like a string. In addition, Visual C++ will display the VARIANT type along with its value. GUIDs won't necessarily display as a string of numbers; Visual C++ will attempt to find the name in the registry and display the name instead.
Debugger formatting symbols	Visual C++ sports a whole list of new debugger symbols as shown in the following list: , hr: Displays 32-bit result or error codes as common COM return values like S_OK or E_NOTIMPL. If it can't display a precise error value, then Visual C++ will attempt to turn the return value into a comment like "Not enough memory."

Continued

Table 1-3 *(continued)*

Feature	*Description*
	,mq: Allows you to display memory as four quadwords to compensate for 64-bit registers. ,st: Displays strings in Unicode or ANSI format depending on the Unicode Strings setting in the AUTOEXP.DAT file. ,wc: Displays numeric values as decoded Windows class flags (WC_ constants). ,wm: Displays numeric values as decoded Windows message values (WM_ constants).
Disassembler output	You'll see a lot more information in the disassembled output, including symbolic information (when available).
Load COFF & Exports	You'll find this option on the Debug tab of the Options dialog (accessed using the Tools | Options command). It enables you to load additional debugging information used by compilers other than Visual C++ (which uses CodeView format debug information). For example, Visual Basic provides Common Object File Format (COFF) debug information in the MSVBVM50.DBG file. The Exports option comes into play when no other debugging information (at least that Visual C++ can recognize) is available. Visual C++ will load the contents of the Exports table of each DLL and attempt to convert it into symbolic information for debugging purposes.
Registers	Several new registers will help you find out more about your application. The Thread Information Blocks (TIBs) pseudo-register provides information about the current thread. The ERR pseudo-register displays the error code of the last error that occurred in your application. This register performs the same task as the GetLastError( ) function call. Adding the ,hr modifier to the ERR register tells it to display a 32-bit error code along with a comment like "Not enough memory." The 64-bit MMX register set now appears in both the Watch and Quick Watch windows. You'll see these registers displayed on all x86 machines even if they don't support the MMX instruction set.
Undecorated symbols	In simple terms, this means that you'll no longer see long lists of meaningless garbage characters with a function name hidden somewhere inside. In addition, all function names will include parameter information, when available.

Feature	*Description*
V-Table and Function Pointer display	Function pointers and virtual table entries get displayed as text entries (symbolically), including parameters whenever possible, instead of as hex addresses.

OLE DB Provider Templates

The OLE DB provider templates enable you to develop OLE DB access interfaces, essentially remote connections to a database. The templates don't perform any magic; they only make accessing OLE DB technology easier. OLE DB is Microsoft's latest database technology. It's designed to provide high performance for database management systems (DBMS) that include an OLE DB provider.

Resource Editor Improvements for Databound Controls

The resource editor enables you to use all the latest ADO and OLE DB technology to create connections to your database. You'll need to use the new ADODC datasource control to create the connection. In addition, you must use the newer ADO compatible databound controls to display data. The only exception to this rule is the use of simple bound controls, which you can use interchangeably with both the ADODC and MSRDC (Microsoft Remote Data Control) datasource controls.

Wizard Support for DocObject Containment

The MFC AppWizard contains a new option in the third step — Active Document Container. This option is available only if you choose the Container or Both Container and Server compound document feature options.

Using DocObject containment (enabled with the Active Document Container option) enables you to use a single frame for all of the types of documents that your application supports instead of creating a separate frame for each one. To get a good idea of exactly how this technology works, check out the Microsoft Office Binder. Normally, OLE will allow you to interact with a single object within a compound document. The DocObject containment technology allows you to activate an entire document, including menus, toolbars, and other support features, within a single frame.

WizardBar Performance Improvements

Projects with a lot of dialogs and control (as witnessed by large CLW files) used to slow to a crawl when WizardBar was visible in the dialog editor box. Microsoft has taken steps to improve performance and reduce the impact WizardBar has on IDE performance.

Development Workstation Setup

To get anywhere in this book, you'll have to have a development workstation. This is where you'll install Windows 2000, write your code, and perform any desktop-level testing. For the most part, you'll want to avoid using your regular workstation for development for two reasons. First, there's no guarantee that an application is going to work the first time and you don't want to crash the machine that contains all of your data. Second, you want to create the cleanest possible environment so that you know for sure that any bugs you're seeing are the result of application errors, not compatibility problems.

The version of Windows 2000 you install depends a great deal on personal taste and the number of machines you plan to use. You'll definitely want to install one of the server versions of Windows 2000 if you plan to use only one machine as both a development workstation and a server. However, some of the programming examples in this book will work better and demonstrate more if you use two machines. (I will always assume you have two machines: one with Windows 2000 Professional installed for development purposes and one with Windows 2000 Server installed for the server.) If you are using two machines, you can set the development workstation up using Windows 2000 Professional. The reason you'd want to use this operating system rather than the server version is so you can get a better idea of how things will actually look from the user's perspective. The server versions include a lot of features that the user won't see, and these features could possibly taint the results of any tests you perform.

For a development workstation that you're going to use exclusively for development and not for testing purposes, make sure you get a fast processor, a lot of RAM, and even more hard drive space. My test workstation includes 512MB of RAM, a 450MHz Pentium II Processor, and a 9GB hard drive. This setup worked very well for my needs in creating code for this book—you'll obviously need to increase your hard drive space as you add more features and create more complex applications. I initially tested every application on my development machine, then on the server, and finally on a test workstation. The test workstation is a 166MHz Pentium machine with 64MB of RAM and a 4GB hard drive. The test workstation you use should reflect the standard issue machine for the environment in which the application will perform.

You may not think about the font you use to work on your machine very often, but working with monospaced fonts all day can lead to eye strain, bugs (like when you confuse an l and a 1), and other problems. Microsoft's choice of Courier for a monospaced font hasn't been well received by many users because it doesn't do the job very well. Fortunately, there are alternatives, some of which are free. Paul Neubauer's Web site at: `http://home.bsu.edu/prn/monofont/` talks about how monospaced fonts are used on a typical Windows system and what replacement fonts are available should you decide you really don't like Courier. This Web site even includes reviews of the various fonts so that you can make a good choice the first time around.

I installed the minimum number of user-level features on my machine. However, you'll need to install MSMQ support if you want to work with the MSMQ samples in Chapter 16. Make sure you install browser support as well, or some examples may fail when you execute them.

Install Visual C++ next. I performed a full install of all features. Make sure your installation includes data access support if you plan to work with any of the database examples in the book. The only database provider support you'll need is SQL Server and Access — there aren't any other database engines required for using the book.

In most cases, you'll want to install the latest version of the Platform SDK. The reason is simple: Many of the examples in the book won't compile without the Platform SDK because Visual C++ shipped without the required header and library files. The examples in this book require a minimum of the September 1998 version of the Platform SDK. Anything earlier won't include the required Windows 2000 support (the September 1998 version is a little flaky, but it does work).

Server Setup

You must have a Windows 2000 Server setup to work with most of the examples in this book. The examples were written using the standard server product on a 200MHz Pentium processor machine with 128MB of RAM. I would recommend a minimum of 4GB of hard drive space, though more is certainly better considering how much space you'll use for the various programming language additions. I purposely used an older technology machine and tested for the minimum environment that would work with the examples in this book. You won't be able to run the server and associated components with anything less than this configuration.

The test server will require access to a number of Windows-specific components if you want the examples in the book to work. The following list summarizes the components that I installed while writing this book:

- Internet Information Server (Complete)
- Management and Monitoring Tools (All)
- Message Queuing Services
- Microsoft Indexing Service
- Networking Services (All)
- Terminal Services (optional, but good to have)

Once you have your test server up and running, there are some additional programs you'll need in order to work with some of the examples in the book. The number of features that you install depends on which examples you want to work

with. Obviously, you'd need a database manager to work with the database examples. The following application list assumes that you want to work with all of the examples.

- ✦ Microsoft Front Page Server Extensions
- ✦ Microsoft Posting Acceptor 2.0
- ✦ Microsoft Visual Studio Enterprise Edition (Server Components)
- ✦ Microsoft Visual Studio Analyzer Server (Part of Visual Studio)
- ✦ Remote Machine Debugging (only if you want to debug your server from a remote location)
- ✦ SQL Server Debugging (Part of Visual Studio)
- ✦ Visual InterDev Server (Part of Visual Studio)
- ✦ Visual SourceSafe Server (Part of Visual Studio)
- ✦ VSEE APE Server (Part of Visual Studio)

Summary

This chapter provided you with an overview of the various kinds of application programming environments you'll need to deal with. We've also spent some time delving into what you can expect from Microsoft's DNA Framework. In addition to programming environment considerations, we also talked about Visual Studio 6. Since this book concentrates on working with Visual C++ 6, this chapter concentrated on that product and didn't provide more than an overview of other Visual Studio products. Finally, we looked at what you'll need in the way of a workstation and server in order to work with all of the examples in this book. This workstation and server setup is likely the minimum you can use to perform most types of programming. Here are some additional conclusions you can draw from the contents of this chapter:

- ✦ There are four arenas of application development that developers normally work with in today's computing environment: desktop, server, Internet/intranet, and large project. Each arena has its own noncoding problems that you'll need to solve as part of creating the application.
- ✦ We will look at the development of the various components that you'll need for a large project.
- ✦ DNA Architecture makes two major contributions from a developer perspective. First, it encourages team development by providing the tools needed to do the job and clearly defining the boundary between various application elements. Second, it provides a road map that demonstrates how the various Microsoft products fit into the big picture.

- DNA will also make the manager's job easier. The most obvious advantage is that DNA reduces the cost of administering applications. Using the DNA Architecture will also make it possible to create a single version of an application that works equally well in all situations. Finally, the DNA Architecture solves some application problems of the past such as disconnected computing.
- Make sure you look for all of the features that DNA can provide. Microsoft hasn't limited itself to the general market in this case — you'll find special versions of DNA designed for vertical markets like manufacturing as well.
- Windows 2000 provides a variety of new services that makes implementing DNA much easier. These services can be placed into the following categories: security, directory, transactions, message queuing, database management, data access, e-mail and collaboration, scripting, and systems management.
- Services alone won't provide everything you need to create applications that rely on DNA; you need tools as well. Microsoft has provided new tools that enable you to create applications quickly (RAD), are modular (COM+), enforce team development, support content creation and publishing, and help you develop Web sites.
- Visual C++ provides a number of new features. Although some of these features fill holes in the environment, many make application development faster or simpler. In some cases, these features are DNA specific. For example, you'll find support for ADO in the latest version of the product.
- Setting up a workstation includes looking at ergonomic details like the kind of font used to display text as well as the hardware and software details.

✦ ✦ ✦